```
     (FILE 'HOME' ENTERED AT 09:49:08 ON 01 JUL 2001)

     FILE 'USPATFULL' ENTERED AT 09:49:13 ON 01 JUL 2001

     FILE 'INPADOC' ENTERED AT 09:49:15 ON 01 JUL 2001

     FILE 'USPATFULL' ENTERED AT 09:49:19 ON 01 JUL 2001
L1           89 S ANNOTATION(P)(E-MAIL OR EMAIL OR (ELECTRONIC(A)(MAIL OR MESSAGES)))
L2          694 S (ANNOTATION OR NOTE OR COMMENT)(P)(E-MAIL OR EMAIL OR (ELECTRONIC(A)(MAIL OR MESSAGES)))
L3           97 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR
RE-EDITING)(P)(ANNOTATION OR NOTE OR COMMENT)(5
L4           77 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR
RE-EDITING)(P)(ANNOTATION OR NOTE OR COMMENT)(3
L5            2 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR
RE-EDITING)(A)(ANNOTATION OR NOTE OR COMMENT)(3
L6           10 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR
RE-EDITING)(A)(ANNOTATION OR NOTE OR COMMENT) A
L7            4 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR
RE-EDITING)(A)(ANNOTATION OR NOTES OR COMMENT)
L8            1 S US6081829/PN
L9            1 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
L10           0 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
L11           0 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
L12           1 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
L13           1 S (ANNOTATION)(2A)(DATABASE OR SERVER)(5A)(PARSE OR PARSING OR EXTRACT? OR COPY)
L14           0 S US5822539/PN AND (ANNOTATION AND (EMAIL OR E-MAIL))
L15           0 S US5822539/PN AND (ANNOTATION AND (EMAIL OR E-MAIL OR MAIL))

     FILE 'USPATFULL' ENTERED AT 10:42:44 ON 01 JUL 2001

     FILE 'PCTFULL' ENTERED AT 10:42:49 ON 01 JUL 2001
L16          51 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
L17           5 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
```

```
     FILE 'EUROPATFULL' ENTERED AT 10:47:30 ON 01 JUL 2001
L18            0 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(A)(ANNOTATION OR NOTES OR CO
L19            0 S (MODIFYING OR MODIFY OR CHANGE OR CHANGING OR CORRECT OR CORRECTING OR RE-EDITING OR
NEW)(5A)(ANNOTATION OR NOTES OR C
L20            0 S (EDIT? OR NEW)(5A)(ANNOTATION OR NOTES OR COMMENT)(10A)(MAIL OR EMAIL OR E-MAIL)

     FILE 'PCTFULL' ENTERED AT 10:49:53 ON 01 JUL 2001
L21            2 S (EDIT? OR NEW)(5A)(ANNOTATION OR NOTES OR COMMENT)(10A)(MAIL OR EMAIL OR
E-MAIL)(5A)(SERVER OR DATABASE)

     FILE 'USPATFULL' ENTERED AT 10:51:18 ON 01 JUL 2001
L22            1 S (EDIT? OR NEW)(5A)(ANNOTATION OR NOTES OR COMMENT)(10A)(MAIL OR EMAIL OR
E-MAIL)(5A)(SERVER OR DATABASE)
L23            1 S US6105055/PN AND (E-MAIL AND SERVER AND ANNOTATION AND DATABASE AND NEW AND REPLY)
                 SET LINELENGTH 250
```

```
L9   ANSWER 8 OF 13  USPATFULL
PI     US 5974449  19991026
PI     US 5974449  19991026
PA     Carmel Connection, Inc., Fremont, CA, United States (U.S. corporation)
AI     US 1997-853290  19970509 (8)
DETD   In the presently preferred embodiment of the present invention, the
       pointer 1156 is a universal resource locator (URL) that points
       to the message included in the web page. The URL is included
       within an HTML file which is included with the notification message 1158
       as an attachment file. This permits an intended recipient to
       access the message presented in the web page simply by launching the
     email attachment file with a messaging interface that
       provides a web browser.
DETD   In the presently preferred embodiment of the present invention, the
       pointer 1730 is a universal resource locator (URL) that points
       to a message included in a web page 1726. The URL is included
       within an HTML file which is included with notification message 1732 as
       an attachment file 1734. This permits an intended recipient to
       access the message such as digitized audio message 1718, presented in
       web page 1726 simply by launching the attachment file 1734
       with a messaging interface such as a computer having a web browser 1736.
       Web page 1726 is configured to present the recipient with options for
       controlling access to digitized audio message 1718 over a suitable
       network such as the Internet so long as the messaging interface can
       support the accessing of digitized audio. This provides a subscriber the
       convenience of sending a voice message to anyone who has an
     email account and access to the Web through a web browser.
```

```
L31  ANSWER 1 OF 1  USPATFULL                                <--
PI      US 6009462  19991228
TI      Replacing large bit component of electronic mail (e-
     mail) message with hot-link in distributed computer
        system                                               <--
PI      US 6009462  19991228
AB      A computer implemented method for down-loading mail
     messages in a distributed computer system. The distributed
     mail service system includes a plurality of client computers
        connected to a mail service system via a network. Mail
     messages are stored in message files of the
     mail service system, a particular mail message
        includes a primary component encoded in a first format, and at least one
        secondary component encoded in a second format different than the first
        component. The particular mail message is requested
        by a particular one of the plurality of client computer systems. The
        secondary component is replaced with a hot-link. The primary component
        and the hot-link are sent to the particular client computer.
SUMM    The present invention relates generally to electronic mail,
        and more particularly to electronic mail messaging in a
        distributed computer system.

SUMM    With the advent of large scale distributed computer systems such as the
        Internet, the amount of information which has become available to users
        of computer systems has exploded. Among this information is electronic
      mail (e-mail). With the improvements in
        means for composing and distributing written messages, the
        amount of e-mail traffic on the Internet has surged.
        It is not unusual for an active Internet user to be exposed to tens of
        thousands of e-mail messages a year.

SUMM    As an advantage, the Internet allows users to interchange useful
        information in a timely and convenient manner. However, keeping track of
        this huge amount of information has become a problem. As an additional
        advantage, the Internet now allows users to exchange information in a
        number of different presentation modalities, such as text, audio, and
        still and moving images. Adapting e-mail systems to
        organize such complex information, and providing efficient means to
```

ck2-4c32-[illegible]

coherently retrieve the information is not trivial.

SUMM As a disadvantage, Internet users may receive junk-**mail** whenever they send to mailing lists or engage in news groups. There are numerous reported incidents where specific users have been overwhelmed by thousands of unwanted **mail messages**. Current filtering systems are inadequate to deal with this deluge.

SUMM Known distributed systems for composing and accessing **e-mail** are typically built around protocols such as Internet Messaging Access Protocol (IMAP), Post Office Protocol (POP), or Simple **Mail** Transfer Protocol (SMTP). Typically, users must install compatible user agent software on any client computers where the **mail** service is going to be accessed. Often, a significant amount of state information is maintained in the users' client computers. For example, it is not unusual to store the entire **mail** database for a particular user in his desk-top or lap-top computer. Normally, the users explicitly organize **mail messages** into subject folders. Accessing **mail** generally involves shipping entire **messages** over the network to the client computer.

SUMM Such systems are deficient in a number of ways. Most computers that a user will encounter will not be configured with user agents compatible with the user's **mail** service. Often, a user's state is captured in a specific client computer which means that work cannot proceed when the user moves to another computer. Managing large quantities of archival **mail messages** by an explicit folder organization is difficult for most users. Accessing **mail** over a low bandwidth network tends to unsatisfactory.

SUMM Therefore, it is desired to provide a **mail** system that overcomes these deficiencies.

SUMM Provided is a computer implemented method for down-loading **mail messages** in a distributed computer system. The distributed **mail** service system includes a plurality of client computers connected to a **mail** service system via a network.

SUMM **Mail messages** are stored in **message** files of the **mail** service system. A particular **mail message** includes a primary component encoded in a first format, and at least one secondary component encoded in a second format different than the first component.

SUMM The particular **mail message** is requested by a particular one of the plurality of client computer systems. The secondary component is replaced with a hot-link, and the primary component and the hot-link are sent to the particular client computer.

DRWD FIG. 1 is a block diagram of an arrangement of a distributed **mail** service system which uses the invention;

DRWD FIG. 2 is a block diagram of a **mail** service system of the arrangement of FIG. 1;

DRWD FIG. 4 is a block diagram of **message** and log files maintained by the system of FIG. 2;

DRWD FIG. 5 is a flow diagram of a parsing scheme used for **mail** **messages** processed by the system of FIG. 2;

DRWD FIG. 6 is a block diagram of a full-text index for the **message** files of FIG. 4;

DRWD FIG. 7 is a diagram of a labeled **message**;

DRWD FIG. 10 is a block diagram for a Multipurpose Internet **Mail** Extensions (MIME) filter.

DETD In FIG. 1, an arrangement 100 provides a distributed **mail** service having features according to the invention. In FIG. 1, one or more client computers 111-113 are connected via a network 120 to a **mail** service system 200 described in greater detail below.

DETD The functionality of the browser 115 can be extended by forms, applets, and plug-ins generally indicated by reference numeral 116. In the preferred embodiment, the browser extensions are in the form of client **mail** application programs described in greater detail below. The client **mail** application programs are downloaded over the network 120 from the **mail** service system 200. The extensions can be implemented using HyperText Markup Language (HTML), JavaScript, Java applets, Microsoft ActiveX, or combinations thereof to provided maximum portability.

DETD A URLs specifies the exact location of a Web-based resource such as a server or data record. The location can include domain, server, user, file, and record information, e.g., HTTP://www.digital.com/.about.userid /file.html/.about.record" An Internet service can be used to send and receive **mail** **messages**. For example, a **mail** **message** can be sent **mail** to the address "jones@ **mail**. digital.com" using the SMTP protocol. As an advantage, the Internet and the Web allow users, with only minor practical limitations, to exchange data no matter where they are using any type of computer equipment.

DETD The **mail** service system 200 includes one or more server computers. Usually, the system 200 is part of some private network (intranet) connected to the public network 120. Typically, an intranet is a distributed computer system operated by some private entity for a selected user base, for example, a corporate network, a government network, or some commercial network.

DETD **Mail** Service System

DETD The **mail** service system 200 can be implemented as one or more server computers connected to each other either locally, or over large geographies. A server computer, as the name implies, is configured to execute server software programs on behalf of client computers 111-113. Sometimes, the term "server" can mean the hardware, the software, or both because the software programs may dynamically be assigned to different servers computers depending on load conditions. Servers typically maintain large centralized data repositories for many users.

DETD In the **mail** system 200, the servers are configured to maintain user accounts, to receive, filter, and organize **mail messages** so that they can readily be located and retrieved, no matter how the information in the **messages** is encoded.

DETD During operation of the arrangement 100, users of the client computers 111-112 desire to perform **e-mail** services. These activities typically include composing, reading, and organizing **e-mail messages**. Therefore, the client computers can make connections to the network 120 using a public Internet service provider (ISP) such as AT&T or Earthlink. Alternatively, a client computer can be connected to the Internet at a "cyber-cafe" such as Cybersmith, or the intranet itself via a local area network. Many other connection mechanism can also be used. Once a connection has been made, a user can perform any **mail** service.

DETD As an advantage, structural and functional characteristics of the arrangement 100 include the following. **Mail** services of the system 200 are available through any Web-connected client computer. The users of the services can be totally mobile, moving among different clients at will during any of the **mail** activities. Composition of a **mail message** can be started on one client, completed on another, and sent from a yet another computer.

DETD These characteristics are attained, in part, by never locking a user's state in one of the client computers in case access is not be possible at a later time. This has the added benefit that a client computer's local storage does not need to be backed-up because none of the important data reside there. In essence, this is based on the notion that the operating platform is the Web, thus access to **mail** service system via the Web is sufficient to access user data.

DETD The service system will work adequately over a wide range of connectivity bandwidths, even for **mail messages** including data in the form of multi-media. **Message** retrieval from a large repository is done using queries of full-text index without require a complex classification scheme.

DETD **Mail** Service System

DETD As shown in FIG. 2, the **mail** service system includes the following components. The system 200 is constructed to have as a front-end a Web server 210. The server 210 can be the "Apache" Web server available from the WWW Consortium. The Web server 210 interacts with a back-end common gateway interface (CGI) programs 220. The programs interface with an account manager 300, a STMP **mail** server 240, and an index server 250. The CGI programs 220 are one possible mechanism. The programs could also be implemented by adding the code directly to the Web server 210, or by adding extensions to the Netscape Server Application Programming Interface (NSAPI) from Netscape.

DETD The top-level functions of the system 200 include send **mail** 241, receive **mail** 242, query index 243, add/remove label to/from **mail** 244, and retrieve **mail** 245. Different servers can be used for the processes which implement the functions 241-245.

DETD The account manager 300 maintains account information. The **mail** server 240 is used to send and receive **mail messages** to and from other servers connected to the network. The index server 250 maintains **mail messages** in **message** files 400, and a full-text index 500 to **messages**. The CGI programs 220 also interact with the **messages** files 400 via a filter 280 for **mail message** retrieval.

DETD The Web server 210 can be any standard Web server that implements the appropriate protocols to communicate via the network using HTTP protocols 201, for example the Apache server. The CGI back-end programs 220 route transactions between the Web server 210 and the operational components of the **mail** service system The CGI back-end 220 can be implemented as C and TCL programs executing on the servers.

DETD As shown in FIG. 3, the account manager 300 maintains account information 301-303 for users who are allowed to have access to the **mail** system 200. Information maintained for each account can include: **mail**-box address 310, e.g., in the form of a "Post Office Protocol (POP-3) address, user password 320, label state 330, named queries 340, filter queries 350, query position information 360, user preferences 370, and saved composition states 380. The full meaning and use of the account information will be come apparent as other components of the system 200 are described.

DETD As an introduction, passwords 320 are used to authenticate users. Labels 330 are used to organize and retrieve **mail messages**. Labels can be likened to annotated notes that can be added and removed to **messages** over their lifetimes, in other words labels are mutable. Labels help users organize their **messages** into subject areas. At any one time, the label state captures all labels that are active for a particular user. Labels will be described in greater detail below.

DETD In the system 200, **mail messages** are accessed by using queries. This is in contrast to explicitly specifying subject folders as are used in many known **mail** systems. A query is composed one or more search terms, perhaps connected by logical operators, that can be used to retrieve **messages**. By specifying the name of a query, a user can easily retrieve **messages** related to a particular topic, phrase, date, sender, etc. Named queries 340 are stored as part of the account information.

DETD Some queries can be designated as "filter" queries 340. This allows a user to screen, for example, "junk **mail**," commonly known as spam. Filter queries can also be used to pre-sort **messages** received from particular mailing lists. Query position information records which **message** the user last selected with a query. This way the user interface can position the display of **messages** with respect to the selected **message** when the query is reissued. User preferences 370 specify the appearance and functioning of the user interface to the **mail** service as implemented by the extended browser 116 of FIG. 1. Saved composition states 380 allow a user to compose and send a **message** using several different client computers while preparing the **message**.

DETD **Mail** Server

DETD Now with continued reference to FIG. 2, the **mail** server system 200 receives (242) new **mail messages** by communicating with the **mail** server 240 using the POP-3 protocol. **Messages** are sent (241) using the SMTP protocol. The **mail** server 240 is connected to the Internet by lines 249. The appropriate routing information in the **mail** server 240 for a particular user can be generated after the user's account has been generated. A "POP Account Name" should be specified as the user's name. In most systems, the name will be case sensitive. The "POP Host" should be the Internet domain name of the **mail** server 240. Here, the case of the letters is ignored. An IP address such as "16.4.0.16" can be used, although the domain name is preferred. In some cases, a particular user's preferred Internet **e-mail** address may be unrelated to the POP Account Name, or the POP Host.

DETD The rapid expansion in the amount of information which is now available on-line has made it much more difficult to locate pertinent information. The question "in which folder did I store that **message**?," becomes more difficult to answer if the number of **messages** that one would like to save increases over long time periods to many thousands. The importance and frequency of accessed **messages** can vary.

DETD Traditionally, the solution has been to structure the **mail messages** in a hierarchical manner, e.g., files, folders, sub-folders, sub-sub-folders, etc. However, it has been recognized that such structures do not scale easily because filing strategies are not consistent over time. Many users find that hierarchical structures are inadequate for substantial quantities of **e-mail messages** accumulated over many years. Particularly, since the meaning and relation of **messages** changes over time. Most systems with an explicit filing strategy require constant and tedious attention to keep the hierarchical ordering consistent with current needs.

DETD **Message** Repository

DETD **Messages** are stored in **message** files 400 and a full-text index. The organization of the **message** files is first described. This is followed by a description of the full-text index 500. As a feature of the present invention, user interaction with the **mail messages** is primarily by queries performed on the full-text index 500.

DETD As shown in FIG. 4, the index server 250 assigns each received **message** 401-402, a unique identification (MsgID) 410. The MsgID 410 is composed of a file identification (FileID) 411, and a **message** number (MsgNum) 412. The FileID "names," or is a pointer to a specific **message** file 420, and the MsgNum is some arbitrary numbering of **messages** in a file, e.g., an index into the file 420.

DETD A **message** never changes after it has been filed. Also, the MsgID 410 forever identifies the same **message**, and is the only ID for the **message**. In the referenced **message** file 240, a **message** entry 430 includes the MsgNum stored at field 431, labels 432, and the content of the **message** itself in field 433.

DETD The number of separate files 240 that are maintained for storing **messages** can depend on the design of the underlying file system and specific implementation details. For example, the size and number of entries of a particular file may be limited by the file system. Also, having multiple files may facilitate file maintenance functions such as back-up and restore.

DETD Although a **message** may never change, the set of labels associated with a **message** may change. Because labels can change, a transaction log 440 is also maintained. The log 440 includes "add" entries (+label) 450, and "remove" entries (-label) 460. Each entry includes the MsgID 451 or 453 of the effected **message** entry, and label that is being added (452) or deleted (453). The contents of the log 440 are occasionally merged with the **message** files 240. Merged entries are removed from the log 440. The label log 440 allows for the mutation of labels attached to data records such as **mail messages**, where the labels and the data which are labeled are stored in the same index.

DETD FIGS. 5 and 6 show how the index server 250 generates the full-text index 500. Newly received **mail messages** are processed in batches 403-404. **Messages** 401 and 402 of a batch are parsed into individual words 510. A batch 403 in a large **mail** service system may include hundreds or thousands of **messages**. The words of the **messages** are parsed in the order that they are received in a batch. Each word is arbitrarily assigned a sequential location number 520.

DETD For example, the very first word of the very first **message** of the very first batch is assigned location "1," the next word location "2," and the last word location "3." The first word of the next **message** is assigned the next sequential location "4," and so forth. Once a location has been assigned to a word, the assignment never changes. If the location is expressed as a 64 bit number, then it is extremely unlikely that there will ever be an overlap on locations.

DETD As the **messages** are parsed, the indexing process generates additional "metawords" 530. For example, an end-of-**message** (eom) metaword is generated for the last word of each **message**. The metawords are assigned the same locations as the words which triggered their generation. In the example shown, the location of the first eom metaword is "3," and the second is "5."

DETD Other parts of the **message**, such as the "To," "From," "Subject," and "Date" fields may generate other distinctive metawords to help organize the fill-text index 500. Metawords help facilitate searches of the index. Metawords are appended with predetermined characters so that there is no chance that a metaword will ever be confused with an actual parsed word. For example, metawords include characters such as "space" which are never allowed in words. Hereinafter, the term "words" means both actual words and synthesized metawords.

DETD After a batch of **messages** have been parsed, the words and their assigned locations are sorted 540, first according to the collating order of the words, and second according their sequential locations. For example, the word "me" appears at locations "3" and "5" as shown in box 550. The sorted batch 550 of words and locations is used to generate the index. Each sorted batch 550 is merged into the index 500, initially empty.

DETD FIG. 6 shows the logical structure of an index 600 according to the preferred embodiment. The index includes a plurality of word entries 610. Each word entry 610 is associated with a unique "word," that appeared at least once in some indexed **message**. The term "word" is used very loosely here, since the parsing of the words in practice depends on which marks/characters are used as word separators. Words do not need to be real words that can be found in a dictionary. Separators can be spacing and punctuation marks.

DETD The indexer 250 will parse anything in a **message** that can be identified as a distinct set of characters delineated by word separators. Dates are also parsed and placed in the index. Dates are indexed so that searches on date ranges are possible. In an active index there may well be millions of different words. Therefore, in actual practice, compression techniques are extensively used to keep the files to a reasonably size, and allow updating of the index 500 as it is being used. The details of the physical on-disk structure of the index 600, and the maintenance thereof are described in U.S. Pat. No. 5,745,899, entitled "A Method for Indexing Information of a Database", issued to Michael Borrows on Apr. 28, 1998, incorporated in its entirety herein by reference.

DETD Labels provide a way for users to annotate **mail** **messages**. Attaching a label to a **message** is similar to affixing a note to a printed document. Labels can be used to replace the folder mechanisms used by many prior art **mail** systems. However, a single **mail message** can be annotated with multiple labels. This compares favorably to folder-based systems where a **message** can only be stored in a single folder.

DETD As shown in FIG. 6, labels are stored in a data structure 650 that parallels and extends the functionality of full-text index 500. Labels are subject to the same constraints as index words. Also, queries on the full-text index 500 can contain labels, as well as words, as search terms. A label is added to a **mail message** by adding a specific index location (or locations) within the **message** to the set of locations referred to by the specified label. Label removal is the opposite. Operations on labels are much more efficient than other operations that mutate the state of the full-text index.

DETD As shown in FIG. 7, a **message** 700 includes a header 701 and a body. The header 701 typically includes the "To", "From", "Date" and "Subject" fields. The header may also include routing information. The body 702 is the text of the **mail message**.

DETD Each **mail message** can initially receive two labels, "inbox" 710 and "unread" 720. **Messages** labeled as "unread" 720 have not yet been exposed for reading. **Messages** with the "inbox" label 710 are deemed to require the user's attention. As will be described below, it possible for **messages** to be labeled as unread but not have the inbox label. These less important **messages** can be read by the user as needed.

DETD Outputting, e.g., displaying or printing, a **message** removes the unread label 720 under the assumption that it has been read. A user can explicitly add or remove the unread label. A **message** can be deleted by attaching a "delete" label 730. This has the effect that the **message** will not been seen again because **messages** labeled as deleted are normally excluded during searches. Removing the deleted label has the effect of "un-deleting" a **message**.

DETD Although a preferred embodiment uses labels for data records that are **mail messages**, it should be understood that "mutable" labels can also be used for other types of data records. For example, labels which can be added and removed can be used with data records such as Web-pages, or news group notes. The key feature here being that labels are indexed in the same index as the record which they label, and that labels can be added and removed.

DETD After **e-mail messages** have been indexed and labeled, the **messages** can be retrieved by issuing full-text queries. A query searches for **messages** that match on words and labels specified in the query. This is in contrast with known **mail** systems where users access **mail** by remembering in which file, folder, or sub-folder **messages** have been placed so the folder can be searched. As an advantage of the present system, users only need to recall some words and labels to find matching

messages.

DETD The syntax of the query language is similar as described in the Burrows reference. A query includes a sequence of primitive query terms, combined by operators such as "and," "or," "not," "near," and so forth. A primitive term can be a sequence of alpha-numeric characters, i.e., a "word," without punctuation marks. If the terms are enclosed without quotation marks ("), the search is for an exact match on the quoted string. A term can be a label. A term such as "from:fred" searches for **messages** with the word "fred" in the "from" field of a **message** header. Similar queries can be formulated for the "to," "from, "cc," and "subject" fields of the header.

DETD A term such as "11/2/96-25/Dec/96" searches for all **messages** in the specified date range. The parsing of dates is flexible, e.g, 12/25/96, 25/12/1996, and Dec/25/96 all mean the same date. In the case of ambiguity (2/1/96) the European order (day/month) is assumed.

DETD During normal operation, the CGI program 220 modifies each issued query by appending a term which excludes the "deleted" label, e.g., "and not deleted." This has the effect of hiding all deleted **messages** from the user of the client. There is an option in the user interface which inhibits this effect to make deleted **messages** visible.

DETD Queries can be "named." Named queries are maintained by the account manager 300. By specifying the name of a query, users can quickly perform a search for **e-mail messages** including frequently used terms. Users can compose complex queries to match on some pattern in indexed **messages**, perhaps intermixing conditions about **messages** having particular text or labels, and to keep the query for subsequent use.

DETD Named queries can be viewed as a way for replacing prior art subject folders. Instead of statically organizing **messages** into folders according to predetermined conditions, queries allow the user to retrieve a specific collection of **messages** depending on a current set of search terms. In other words, the conditions which define the collection are dynamically expressed as a query.

DETD Recently performed queries are kept in a "history" list. Accordingly, frequently performed queries can readily be re-issued, for example, when the index has been changed because of newly received **mail**, or because of actions taken by other client computers.

DETD Queries can also be used to perform the function of prior art "address books." In many known **e-mail** systems, users keep address books of frequently used addresses. From time to time, users can add and remove addresses. There, the address books are statically maintained as separate data structures or address book files. For example, there can be "personal" and "public" related address books. In contrast, here, there is no separately stored address book. Instead, an "address book" is dynamically generated as it is needed. The dynamic address book is generated from the files 400 and the full-text index 500 as follows.

Gupta et. al.

L33 ANSWER 1 OF 1 USPATFULL
PI US 6009462 19991228 <--
AI US 1997-876605 19970616 (8)
PA Digital Equipment Corporation, Maynard, MA, United States (U.S. corporation)
PI US 6009462 19991228 <--

DETD Although a message may never change, the set of labels associated with a message may change. Because labels can change, a transaction log 440 is also maintained. The log 440 includes "add" entries (+label) 450, and "remove" entries (-label) 460. Each entry includes the MsgID 451 or 453 of the effected message entry, and label that is being added (452) or deleted (453). The contents of the log 440 are occasionally merged with the message files 240. Merged entries are removed from the log 440. The label log 440 allows for the mutation of labels **attached** to data records such as mail messages, where the labels and the data which are labeled are stored in the same index.

DETD Labels provide a way for users to annotate mail messages. **Attaching** a label to a message is similar to affixing a note to a printed document. Labels can be used to replace the folder mechanisms used by many prior art mail systems. However, a single mail message can be annotated with multiple labels. This compares favorably to folder-based systems where a message can only be stored in a single folder.

DETD Outputting, e.g., displaying or printing, a message removes the unread label 720 under the assumption that it has been read. A user can explicitly add or remove the unread label. A message can be deleted by **attaching** a "delete" label 730. This has the effect that the message will not been seen again because messages labeled as deleted are normally excluded during searches. Removing the deleted label has the effect of "un-deleting" a message.

DETD The system also attempts to detect components in messages, such as explicitly "**attached**" or implicitly "embedded" files. The files can be in any number of possible formats. The content of these files are displayed by the browser 115. The specific display actions used will depend on how the browser is configured to respond to different component file formats.

DETD For some file formats, for example Graphics Interface Format (GIF) and Joint Photographic Experts Group (JPEG), the component can directly be displayed. It is also possible to configure the browser with a "helper" applet to "display" **attached** files having specific format types as "icons." For example, the message may be in the form of an audio message, in which case, the message needs to be "said," and not displayed. For some message formats, the browser may store some of the content in file system of the client computer.

DETD Because the mail service system 200 allows mail messages to include **attached** or embedded multi-media files, mail messages can become quite large. In the prior art, the entire mail message, included files are typically shipped to the client computer. Thus, any part of the mail message can immediately be read by the user after the message has been received in the client.

DETD As shown in FIG. 10, the mail service system 200 can recognize messages components that are included as such. The system 200 can discover an explicitly **attached** file 1010 to a message 1000, and the system 200 can also heuristically discover textual components 1021-1021 that are implicitly embedded without MIME structuring in the message. For example, the system 200 can recognize embedded "uuencoded" enclosures, base 64 enclosures, Postscript (and PDF) documents, HTML pages, and MIME fragments.

DETD Accordingly, the system 200 is configured to "hold-back" such components 1010, 1020-1021 encoded in different formats using a "MIME" filter 1001. The **attached** and embedded components are replaced by hot-links 1031 in a reduced size message 1030. Only when the user clicks on one of the hot-links 1031 is the components sent to the requesting client computer.

DETD The Name Current Query allows a user to **attach** a text string to the current query. This causes the system 200 to place the query in the account for the user for subsequent use. The Forget Named Query option deletes a named query.

DETD Forward: This function sets up a window for composing, a new message. A selected message is **attached** to the new message. The **attached** messages are forwarded without the need of down-loading the messages to the client computer.

DETD Send: Sends a message. Any **attachments** are included before sending the message. The user is notified of invalid recipients by a status message, and editing of the message can continue. Otherwise, the window is switched to read-only mode.

CLM What is claimed is:
10. The method of claim 1, wherein: the secondary component is **attached**.

DETD As shown in FIG. 8, a user of a client computer 820 can generate address book type information using a form 800 supplied by one of the client **mail** application programs 116. The form 800 includes, for example, entry fields 801-803 for address related information such as name, phone number, (hard-copy) **mail** address, and (soft-copy) **e-mail** address, and so forth. Alternatively, address information can be selected from a prior received **mail message** 805 by clicking on appropriate fields in the header or body of the **message** 805.

DETD From the perspective of the **mail** service system 200 and the index server 250, the address book information is handled exactly as a received **mail message**. This means that, for example, the data of the fields 801-803 are combined into an "address book" **mail message** 810. An "address" label 809 can also be added to the entry using the labeling convention as described herein. The address book **mail message** 810 and label 809 can be stored in one of the **message** files 400. Additionally the **message** 810 can be parsed and inserted into the full-text index 500 as are the words and labels of any other **mail message**. In other words, the address information of form 800 is merged and blended with the full-text index 500.

DETD After the address information has been filed and indexed, the address information can be retrieved by the user of the client computer 820 composing a query 830 using the standard query interface, with perhaps, the label "address" as one of the query terms. The exact content to be retrieved is determined at the time that the terms and operator of the query 830 are composed by the user. The address information, i.e., one or more address book **mail messages**, which satisfies the query is returned to the client computer 820 as the dynamic address book 840. The user can then select one of the addresses as a "to" address for a new, reply, or forward **mail message**.

DETD **Message** Resemblance

DETD It is also possible to search for **messages** which resemble a currently selected **message**. In this case a document resemblance technique can be used. Such a technique is described in U.S. patent application Ser. No. 08/665,709 entitled "A Method for Determining the Resemblance of Documents", filed by Broder et al. on Jun. 18, 1996, incorporate in its entirety herein by reference. This allows a user to find all **messages** which closely relate to each other.

DETD When a search for an issued query completes, the results of the search are presented in an order according to their MessageID 411, FIG. 4. In practice, this means that qualifying **messages** are presented in the temporal order of when the **messages** were received.

DETD Most prior art **e-mail** systems allow other sort orders, such as by sender, or by **message** thread (a sequence of related **messages**). There is no need for such capabilities here. Consider the following possibilities.

DETD **Messages** from a particular user can be specified by including in a query a term such as "from:jones." This will locate only **messages** from a particular user. You can select **messages** of a particular "thread" by using the "view discussion" option of the user interface described below. As stated above, **messages** for a particular date range can be specified in the query.

DETD Filtering **Messages**

DETD In order to facilitate **mail** handling, particularly for someone receiving a large amount of **e-mail**, a user can configure the filter 280 to his or her own preferences as shown in FIG. 9. A **message** filter is specified as one or more name "filter" queries 910. The named query 910 is stored as part of the account information of FIG. 3. The named filter query 910 can be composed on a client computer 920 using the client **mail** application programs down-loaded from the **mail** service system 200.

DETD New **messages** 930 received by the **mail** service system 200 are stored, parsed, and indexed in the **message** files 400 and full-text index 500 as described above. In addition, each new **message** 930 can be compared with the named queries 910. If the content of a new **message** 930 does not match any of the named filter queries 910, then the new **message** 930 is given the inbox label 710 and the unread label 720, i.e., the **message** is placed in the "In-box" 940 for the user's attention. Otherwise, the new **message** 920 is only given the unread label 720.

DETD For example, **mail** which is sent out typically has a "from" field including the name of the sender, e.g., "From: Jon Doe," in the **message** header. Alternatively, the body of the **mail** **message** may include the text, "You are getting this **message** from your good friend Jon Doe." The user Jon Doe can set up a named filter query "SentByME" as "From near (Jon Doe)". This query will match any **message** which contains the word "from" near the word phrase "Jon Doe." The effect is that users do not explicitly become aware of **messages** that match on the filter query 910. For example, a user may want to filter **messages** which are "cc" copies to one self. A user may also desire to filter out junk **e** **-mail** **messages** arriving from commercial **e-** **mail** distributors at known domains, or pre-sort **messages** received via mailing lists.

DETD **Message** Display Options

DETD From the user's perspective, access to the **mail** services is implemented by extensions to the Web browser, such as Java applets. **Messages** are normally displayed by their primary component being transmitted to the client in the HTML format, and being displayed in the Java applet's window. The first line of a displayed **message** contains any "hot-links" which the user can click to display the **message** in one of the Web browser's windows, either with the

HTML formatting, or as the original text uninterpreted by the system.

DETD It should be noted, headers in Internet **messages**, depending on routing, can be quite lengthy. Therefore, it is possible to restrict the view to just the "from," "to," "cc," "date," and "subject" fields of the header.

DETD When displaying retrieved **messages**, the system 200 heuristically locates text strings which have the syntax of **e-mail** addresses. If the user click on one of these addresses, then the system will display a composition window, described below, so that the user can easily generate a reply **message** to the selected **e-mail** address(es).

DETD Similarly, when displaying retrieved **messages**, the system 200 heuristically locates text strings that have the syntax of an URL, and makes the string a hot-link. When the user clicks on the hot-link, the URL is passed to the browser, which will retrieve the contents over the network, and process the content in the normal manner.

DETD The system also attempts to detect components in **messages**, such as explicitly "attached" or implicitly "embedded" files. The files can be in any number of possible formats. The content of these files are displayed by the browser 115. The specific display actions used will depend on how the browser is configured to respond to different component file formats.

DETD For some file formats, for example Graphics Interface Format (GIF) and Joint Photographic Experts Group (JPEG), the component can directly be displayed. It is also possible to configure the browser with a "helper" applet to "display" attached files having specific format types as "icons." For example, the **message** may be in the form of an audio **message**, in which case, the **message** needs to be "said," and not displayed. For some **message** formats, the browser may store some of the content in file system of the client computer.

DETD Since the client computers 111-113 may access the **mail** service system via low-bandwidth network connections, an attempt is made to minimize the amount of data that are sent from the **mail** service system to the client computers. Even over high-speed communications channels, minimizing the amount of network traffic can improve user interactions.

DETD Because the **mail** service system 200 allows **mail messages** to include attached or embedded multi-media files, **mail messages** can become quite large. In the prior art, the entire **mail message**, included files are typically shipped to the client computer. Thus, any part of the **mail message** can immediately be read by the user after the **message** has been received in the client.

DETD As shown in FIG. 10, the **mail** service system 200 can recognize **messages** components that are included as such. The system 200 can discover an explicitly attached file 1010 to a **message**

1000, and the system 200 can also heuristically discover textual components 1021-1021 that are implicitly embedded without MIME structuring in the **message**. For example, the system 200 can recognize embedded "uuencoded" enclosures, base 64 enclosures, Postscript (and PDF) documents, HTML pages, and MIME fragments.

DETD Accordingly, the system 200 is configured to "hold-back" such components 1010, 1020-1021 encoded in different formats using a "MIME" filter 1001. The attached and embedded components are replaced by hot-links 1031 in a reduced size **message** 1030. Only when the user clicks on one of the hot-links 1031 is the components sent to the requesting client computer.

DETD The following sections described how the Web browser 115 is configured to provided the **e-mail** services of the system 200.
The functions described can be displayed as pull-down menus, or as button bars depending on a desired appearance. Preferrably, the functions are implemented as Java applets.

DETD This menu includes the View Discussion, Name Current Query, Forget Named Query, Exclude "deleted" **Message**, and Your Query Options. The View Discussion option issues a query for **messages** related to the currently selected **message**. Here, "related" means any **messages** which share approximately the same subject line, and/or being in reply to such a **message**, or **messages** linked by a common standard "RFC822" **message** ID.

DETD The Excluded "deleted" **message** option omits from a query result all **messages** that have the deleted label. This is the default option. Clicking on this option changes the behavior of the system 200 to include, in response to a query, "deleted" **messages**. The Your Named Queries option displays a particular user's set of named queries 340. Clicking on any of the displayed names issues the query.

DETD The client keeps a history of, for example, the last ten queries to allow for the reissue of queries. The options of this menu are Go Back, Redo Current Query, Go Forward, and The History List. Go Back reissues the query preceding to the current query. Redo reissues the current query. This option is useful to process **messages** which have recently arrived, or in the case where the user's actions have altered the **messages** files 400 in some other manner. Go Forward reissues the query following the current query. The History List displays all of the recently issued queries. Any query listed can be reissued by clicking on the query.

DETD **Messages** Menu

DETD Options here include: Select All, Select Unread, Select Read, Mark As Unread, Mark As Read, Add Labels, Remove Labels, and Use Built-in Viewer. The Select All option selects all **messages** which match the current query. The next two options respectively select **message** that do not, and do have the unread label. The following two options add and remove labels label to currently selected

messages.

DETD The user interface normally displays a **message** by converting the **message** to an HTML format and presenting it to an HTML viewer which can either be in the browser's main window, or with a built-in viewer. The last option of the **message** menu selects the viewer.

DETD The help options can be used to display informational pages on how to use the various features of the system. The help pages are down-loaded on demand into the client computer from the **mail** service system 200.

DETD Add: This button is used to add a selected label to a **message**.

DETD Delete: With this button, a deleted label is added to a **message** .

DETD Unlabel: Used to remove a single label mentioned in a query from a **message**.

DETD Next: Selects a next **message**.

DETD Prev: Selects a preceding **message**.

DETD Newmail: Issues a query for all **message** having the inbox label.

DETD **Message** Display Button Bar

DETD Detach: Generate a new top-level window to display selected **messages**.

DETD Compose: Generate a window for composing new **mail** **messages**.

DETD Forward: This function sets up a window for composing, a new **message**. A selected **message** is attached to the new **message**. The attached **messages** are forwarded without the need of down-loading the **messages** to the client computer.

DETD Reply To All: This function sets up a window for composing a new **message** with the same recipients as those in a selected **message**.

DETD Reply To Sender: Set up a window for composing a new **message** to the sender of a selected **message**.

DETD Access to the composition window is gained by clicking on the Compose, Forward, Reply, or Modify button, or by clicking on a "**mail** -to" hot link in a displayed **message**. Compose begins a new **message**, forward is used to send a previously received **message** to someone else, reply is to respond to a **message**, and modify allows on to change a **message** which has not yet been sent. The **mail** service allows a user to compose multiple **messages** at a time.

DETD The text of a **message** is typed in using an available composition window, or generating a window if none are available. The exact form of the typing area of the composition window depends on the nature of the windowing system used on a particular client computer. Typically, while typing the user can use short-cuts for editing actions such as cut, paste, copy, delete, undo, and so forth.

DETD Text portions from another **message** can be inserted by using the Insert Msg, or Quote Msg buttons. If an entire **message** is to be included, then the Forward button should be used. The **message** will not actually be posted until the send function is selected. While the **message** is being composed, it is periodically saved by the **mail** system. Thus, a composition session started using one client computer in an office, can easily be completed some time later using another computer.

DETD Send: Sends a **message**. Any **attachments** are included before sending the **message**. The user is notified of invalid recipients by a status **message**, and editing of the **message** can continue. Otherwise, the window is switched to read-only mode.

DETD Close: After a **message** has been sent, or the discard button is clicked, this button replaces the send button to allow one to close the composition window.

DETD Discard: This button is used to discard the **message** being composed, and switches the window to read-only. A user can then click the close or modify buttons.

DETD Modify: After a **message** has been successfully sent, or if the discard button has been clicked, this button appears in place of the discard button to allow the user to compose another **message** derived from the current **message**.

DETD Insert Msg: Replace the selected text with displayed text from a selected **message**.

DETD Quote Msg: Replace the selected text with displayed text from a selected **message** so that each line is preceded by the ">" character.

CLM What is claimed is:

1. A computer implemented method for down-loading **mail messages** in a distributed computer system, the distributed computer system including a plurality of client computers connected to a **mail** service system via a network, comprising the steps of: storing **mail messages** in **message** files of the **mail** service system, a particular **mail message** including a primary component encoded in a first format, and at least one secondary component encoded in a second format different than the primary component; requesting, by a particular one of the plurality of client computers, a particular one of the stored **mail messages**; recognizing from the particular one of the stored **mail messages** the at least one secondary component as being of the is second format; replacing the secondary component of the particular **mail message** with a hot-link; and sending over a network connection of the network the primary component and the hot-link to the particular client computer but holding back with a filter the secondary component.

8. The method of claim 1 wherein the secondary component includes a

Gupta et. al.

```
    tertiary component in a third format, comprising the steps of;
    requesting, by the particular client computer, the particular
  mail message; replacing the tertiary component with
    another hot-link; and sending the primary component and the other
    hot-link to the particular client computer.


=> d pn, ai

L31  ANSWER 1 OF 1  USPATFULL
PI     US 6009462  19991228                                        <--
AI     US 1997-876605  19970616 (8)

=> d pn, ai , pa

L31  ANSWER 1 OF 1  USPATFULL
PI     US 6009462  19991228                                        <--
AI     US 1997-876605  19970616 (8)
PA     Digital Equipment Corporation, Maynard, MA, United States (U.S.
       corporation)
```

Gupta et. al.

PI US 5884313 19990316 <--

AB When a client computer requests data from a disk or similar device at a server computer, the client exports the memory associated with an allocated read buffer by generating and storing one or more incoming MMU (IMMU) entries that **map** the read buffer to an assigned global address range. The remote data read request, along with the assigned global address range is communicated to the server node. At the server, the request is serviced by performing a memory import operation, in which one or more outgoing MMU (OMMU) entries are generated and stored for **mapping** the global address range specified in the read request to a corresponding range of local **physical addresses**. The mapped local **physical addresses** in the server are not locations in the server's memory. The server then performs a DMA operation for directly transferring the data specified in the request message from the disk to the mapped local **physical addresses**. The DMA operation transmits the specified data to the server's network interface, at which the mapped local **physical addresses** to which the data is transferred are converted into the corresponding global addresses. The specified data with the corresponding global addresses are then transmitted to the client node. The client converts the global addresses in the received specified data into the local **physical addresses** corresponding to the allocated receive buffer, and stores the received specified data in the allocated receive buffer.

SUMM For the purposes of this discussion it is assumed that the NIC is suitable for memory mapped message passing. That is, the NIC must be directly addressable using local **physical addresses**, and direct loads and stores to from and to locations in a remote computer node can be performed through the NIC without having to use the NIC's driver software.

SUMM FIG. 2 shows a simplified representation of a conventional communications interface (or NIC) 60, such the ones used in the computer nodes of FIG. 1, showing only the components of particular interest. The NIC 60 typically includes two address **mapping** mechanisms: an incoming memory management unit (IMMU) 70 and an outgoing memory management unit (OMMU) 72. The purpose of the two memory management units are to **map** local **physical addresses** (PA's) in each computer node to global addresses (GA's) and back.

Transport logic 74 in the NIC 60 handles the mechanics of transmitting and receiving message packets, including looking up and converting addresses using the IMMU 70 and OMMU 72.

SUMM Memory **Mapping** between Virtual, Local Physical and Global Address Spaces

SUMM Referring to FIGS. 3 and 4, the nodes in a distributed computer system (such as those shown in FIG. 1) utilize a shared global address space GA. Each node **maps** portions of its local address space LA into "windows" in the global address space. Furthermore, processes on each of the nodes **map** portions of their private **virtual address** space VA into the local **physical address** space PA, and can furthermore export a portion of the local **physical address** space PA into a window in the global address space GA. The process of "exporting" a portion of the local **physical address** space is also sometimes referred to as "exporting a portion of the local **physical address** to another node," because a specific other computer node is given read and/or write access to the exported portion of the local **physical address** space via an assigned global address space range.

SUMM It should be noted that the local **physical addresses** (e.g., PA1 and PA2) shown in FIGS. 3 and 4 are physical bus addresses and are not necessarily memory location addresses. In fact, many **physical addresses** are actually mapped to devices other than memory, such as the network interface. For example, when physical memory on a first computer is exported to a second computer, the **physical addresses** used in the second computer to write to the exported memory are not mapped to any local memory; rather they are mapped to the second computer's network interface.

SUMM When a message containing a destination address is sent from a process in node A 50 to a process in node B 52, a series of address translations (also called address **mapping** translations) are performed on the destination address. A **virtual address** VA1 from a process in node A is first translated by the TLB (translation lookaside buffer) 80-A in node A's CPU 54-A into a local **physical address** PA1. The local **physical address** PA1 is then translated by the outgoing MMU (OMMU) 72-A in node A's network interface 60-A into a global address GAx. When the message containing the global address is received by node B, the global address GAx is converted by the incoming MMU (IMMU) 70-B in node B's network interface 60-B into a local **physical address** PA2 associated with node B. The local **physical address** PA2 corresponds to a **virtual address** VA2 associated with a receiving process. A TLB 80-B in node B's CPU 54-B **maps** the **virtual address** VA2 to the local address PA2 where the received message is stored.

SUMM It should be noted here that TLBs generally only translate **virtual addresses** into local **physical addresses**, and not the other way around, and thus some of the arrows in FIG. 4 represent **mappings** rather than actual address translations. When the receiving process in the node B reads a received message at address VA2, the TLB 80-B will translate that **virtual address** into the same local address LA2 determined by the network interface's IMMU 70-B as the destination address for the received message.

SUMM Receive buffers are typically allocated in page size chunks, since each MMU entry generally represents a **mapping** of one or more pages (and more generally 2.sup.n pages, for integer values of n.gtoreq.0) of address space. Larger receive buffers, or receive buffers of irregular size, may be constructed using multiple MMU entries by user level protocols. Once the receive buffers are allocated and the corresponding MMU **mappings** are established, user level programs can manage the receive buffers without kernel intervention. Many different kinds of user level message passing "API's" (**application** program interfaces) can be built on top of the basic receive buffer mechanism. This includes the send and receive Unix primitives, **sockets**, ORB (object resource broker) transport, remote procedure calls, and so on. The basic message passing mechanism is designed to be as "light weight" and efficient as possible, so as to take as few processor cycles as possible.

SUMM The present invention utilizes the local **physical address** to global address **mapping** mechanisms discussed above.

SUMM FIG. 5 shows the conventional procedure for a process on node B to read information from a disk at node A. The first step is for Node B to set up a receive buffer by "exporting memory" to Node A (step 80), so that Node A can write a message into it. In some implementations, this step may be performed in advance, because it is known in advance that Node B will be performing many disk reads from Node A. In most implementations, however, the memory exporting step is performed in response to a remote disk read request by a user or kernel process in Node B. The memory exporting step 80 is performed by creating an IMMU entry in Node B that **maps** the **physical address** range of a receive buffer in Node B's memory to a corresponding range of global addresses. As indicated above, Node B will typically have a range of global addresses preassigned to it for exporting memory to other nodes.

However, other mechanisms for assigning global addresses would be equally applicable.

SUMM At the server (Node A), when the request message is received, the server sets up an OMMU entry to import the memory being exported by the requesting client node, Node B, (step 84). The OMMU entry set up at step 84 **maps** a range the global address range specified in the received message to a corresponding range of physical memory in the server node. If necessary (e.g., if insufficient contiguous memory is available and/or the size of the mapped address range is not equal to 2.sup.n pages), the server node will generate two or more OMMU entries so as to **map** the specified global address space to two or more local **physical address** ranges.

SUMM After the server transmits the requested data to the requesting node (steps 86, 88 and 90), the server "**tears** down" the **connection** by deleting the OMMU entry (or entries) for the imported memory associated with the request from Node B.

SUMM In response to the completed message, the requesting node **tears** down its side of the **connection** by deleting the corresponding IMMU entry (or entries) so as to unexport the memory used for the request, and to return local read/write control to that portion of the requesting system's local physical memory (step 94). Then, or in parallel with the **tear** down operation, the requesting node processes the received data (step 96).

SUMM When the client node requests data from a disk or similar device at the server node, the client node exports the memory associated with an allocated read buffer by generating and storing one or more incoming MMU (IMMU) entries that **map** the read buffer to an assigned global address range. The remote data read request, along with the assigned global address range is communicated to the server node.

SUMM At the server node, the request is serviced by performing a memory import operation, in which one or more outgoing MMU (OMMU) entries are generated and stored for **mapping** the global address range specified in the read request to a corresponding range of local **physical addresses**. The mapped local **physical addresses** in the server are not locations in the server's memory. The server then performs a disk controller DMA operation for directly transferring the data specified in the request message from the disk to the mapped local **physical addresses**. The DMA operation transmits the specified data to the server's network interface, at which the mapped local **physical addresses** to which the data is transferred are converted into the corresponding global addresses. The specified data with the corresponding global addresses are then transmitted to the client node.

SUMM The client node responds to receipt of the specified data by converting the global addresses in the received specified data into the local **physical addresses** corresponding to the allocated receive buffer, and storing the received specified data in the allocated receive buffer.

DRWD FIG. 3 depicts virtual, local and global address spaces and **mappings** between those address spaces.

DETD The first step is for Node B to set up a receive buffer by "exporting memory" to node A (step 80). In some implementations, this step may be performed in advance, because it is known in advance that Node B will be performing many disk reads from Node A. In most implementations, however, the memory exporting step is performed in response to a remote disk read request by a user or kernel process in Node B. The memory exporting step 80 is performed by creating an IMMU entry in Node B that **maps** the **physical address** range of a receive buffer in Node B's memory to a corresponding range of global addresses. As indicated above, Node B will typically have a range of global addresses preassigned to it for exporting memory to other nodes. However, other mechanisms for assigning global addresses would be equally applicable.

DETD At the server (Node A), when the request message is received, the server sets up an OMMU entry to import the memory being exported by the requesting client node, Node B, (step 300). The OMMU entry set up at step 300 **maps** a range the global address range specified in the received message to a corresponding range of physical memory in the server node. However, unlike in step 84 in FIG. 5, the mapped local **physical addresses** do not denote locations in the second computer's memory, rather the mapped local **physical addresses** are **physical addresses** reserved for use by the server's network interface. If necessary (e.g., if a sufficiently large contiguous range of **physical addresses** assigned to the network interface is not available and/or the size of the mapped address range is not equal to 2.sup.n pages), the server node will generate two or more OMMU entries so as to **map** the specified global address space to two or more local **physical address** ranges.

DETD Once the IMMU in the requesting node B and the OMMU in the responding or sending node A have been set up, the disk controller in the server sets up its internal DMA 332 (shown in FIG. 8) to copy the requested data to the local **physical address** (in the server) assigned to the imported memory (step 301). In other words, the DMA's source address, destination address, and data quantity count registers are set up with the values required to perform a direct data transfer from the disk device to local **physical addresses** assigned to the memory located in the requested client computer.

DETD After the "open barrier" step, the disk controller DMA transfer operation is initiated, causing the requested data to be transmitted from the disk directly to the server's network interface (304). Because the destination addresses for the DMA match are mapped by the OMMU, the NIC card receives the disk data directly off the relevant internal bus and retransmits that data onto the communication channel to the requesting server with the local physical destination addresses translated into the corresponding global addresses. At the requesting client computer, the global addresses in the transmitted data are converted into local **physical addresses** by the receiving client computer's IMMU, and then the data is transmitted on a local internal bus for storage in the receive buffer corresponding to those local **physical addresses**. As indicated earlier, all the aforementioned address translations and data retransmissions by the network interfaces are totally automatic and represent the standard operation of such devices.

DETD Finally, the server "**tears** down" the **connection** by deleting the OMMU entry (or entries) for the imported memory associated with the request from Node B (step 92).

DETD In response to the completed message, the requesting node **tears** down its side of the **connection** by deleting the corresponding IMMU entry (or entries) so as to unexport the memory used for the request, and to return local read/write control to that portion of the requesting system's local physical memory (step 94). Then, or in parallel with the **tear** down operation, the requesting node processes the received data (step 96).

DETD **application** programs 342, including **application** programs that can request data from a remotely located disk storage device;

DETD a file system 341 that, among other things, handles file transfers between computer nodes in response to **application** program commands; the file system includes client and server file system portions for handling the client and server sides of a file transfer;

CLM What is claimed is:

1. A method of performing a remote disk read operation between first and second computers, comprising the steps of: at the first computer: **mapping** a range of physical local addresses associated with a receive buffer to a corresponding range of global addresses; sending a request message to the second computer, the request message specifying data to be retrieved from the disk located at the second computer and the range of global addresses associated with the allocated receive buffer; at the second computer, responding to receipt of the request message by: **mapping** the range of global addresses specified in the request message to a corresponding range of local **physical addresses**, wherein the mapped local **physical addresses** do not denote locations in a memory in the second

computer's memory; performing a DMA operation for directly transferring the data specified in the request message from the disk to the mapped local **physical addresses**, the DMA operation transmitting the specified data to a network interface in the second computer at which the mapped local **physical addresses** to which the data is transferred are converted into the corresponding global addresses; and transmitting the specified data with the corresponding global addresses to the first computer; and at the first computer, responding to receipt of the specified data by: converting the global addresses transmitted with the received specified data into the local **physical addresses** corresponding to the allocated receive buffer; and storing the received specified data in the allocated receive buffer.

2. A method of performing a remote disk read operation between first and second computers, comprising the steps of: at the first computer, responding to an **application** program request for data from a disk located at the second computer by: allocating a receive buffer in memory in the first computer, the receive buffer having an associated range of local **physical addresses**; storing an entry in an input memory **mapping** unit in the first computer for **mapping** the local **physical address** range associated with the allocated receive buffer to a corresponding range of global addresses; and sending a request message to the second computer, the request message specifying data to be retrieved from the disk located at the second computer and the range of global addresses associated with the allocated receive buffer; at the second computer, responding to receipt of the request message by: storing an entry in an output memory **mapping** unit in the second computer for **mapping** the range of global addresses specified in the request message to a corresponding range of local **physical addresses**, wherein the mapped local **physical addresses** do not denote locations in a memory in the second computer's memory; and performing a DMA operation for directly transferring the data specified in the request message from the disk to the mapped local **physical addresses**, the DMA operation transmitting the specified data to a network interface in the second computer at which the mapped local **physical addresses** to which the data is transferred are converted into the corresponding global addresses; and transmitting the specified data with the corresponding global addresses to the first computer; and at the first computer, responding to receipt of the specified data by: converting the global addresses transmitted with the received specified data into the local **physical addresses** corresponding to the allocated receive buffer; and storing the received specified data in the allocated receive buffer.

3. The method of claim 2, wherein at the second computer the entry in an output memory **mapping** unit is deactivated after the specified data is successfully transmitted to the first computer; and at the first computer the entry in the input memory **mapping** unit is deactivated after the specified data is successfully received from the second computer.

4. In a distributed computer system, apparatus for performing a remote disk read operation between first and second computers, comprising: at the first computer: a CPU; memory, including memory in which a receive buffer is allocated; an input memory management unit (IMMU); a network interface; a network interface driver procedure, executable by the first computer's CPU, that: stores in the IMMU an entry for **mapping** a range of physical local addresses associated with the receive buffer to a corresponding range of global addresses; and a file system procedure, executable by the first computer's CPU, that sends a request message to the second computer via the network interface and a communication channel coupled to the network interface, the request message specifying data to be retrieved from a disk located at the second computer and the range of global addresses associated with the allocated receive buffer; at the second computer: a CPU; memory; the disk that stores the data specified in the request message; an output memory management unit (OMMU); a network interface coupled to the communication channel; a network interface driver, executable by the CPU, that: stores in OMMU an entry that **maps** the range of global addresses specified in the request message to a corresponding range of local **physical addresses**, wherein the mapped local **physical addresses** do not denote locations in a memory in the second computer's memory; and a disk controller, coupled to the disk, wherein the disk controller is coupled to the network interface by an internal bus; the disk controller including DMA logic for directly transferring the data specified in the request message from the disk to the mapped local **physical addresses**, the DMA operation transmitting the specified data to the network interface in the second computer at which (A) the mapped local **physical addresses** to which the data is transferred are converted into the corresponding global addresses in accordance with the entry stored in the OMMU, and (B) the specified data is transmitted with the corresponding global addresses to the first computer via the communication channel; wherein, at the first computer, the network interface includes logic for responding to receipt of the specified data by (A) converting the global addresses transmitted with the received specified data into the local **physical addresses** corresponding to the allocated receive buffer, and (B) storing the received specified data in the allocated receive buffer.

Gupta et. al.

```
L9    ANSWER 3 OF 3      PCTFULL  COPYRIGHT 2001 MicroPatent
PI    WO 9428480          A1 19941208

CLM   Phone: (313) 930-7777
      EMAIL: Callisto@imagine.com
      FAX: (313) 930-7776
      Mail: Imagine Multimedia, Inc.

      m Creating a New Presentation
      I . If Callisto isn't running yet, doubie-click on the Callisto
    icon
      in the File Viewer (or on the dock if you put it there after
      installation). Callisto's Main Menu and Tool Panel appear.

      I
      SUBSTITUTE SHEET (RULE 26)
      BETA R-ELEASE
      0 Opening an Existing Presentation
      1. If Callisto isn't running yet, double-click on the Callisto
    icon
      in the File Viewer (or on the dock if you put it there after
      installation).

      The Services Menu
      The Services menu is used to request the services of another
      application. Some menu selections access standard NeXT
      applications (e.g., Grab, Edit, Mail), but the rest of the
      menu's
      content depends on which applications have been installed on
      your workstation or network.
```

```
L1    ANSWER 1 OF 1  USPATFULL
PI      US 5893087  19990406
CLM     What is claimed is:
```

5. The method of claim 1 wherein said at least one row comprises a template for a structured **e-mail** message.

16. A method for storing and retrieving data in a computer system including a memory, a central processing unit and a display, said method including the steps of: configuring said memory according to an extensible logical table, said extensible logical table including: a plurality of rows, each said row including an object identification number (OID) to identify each said row, each said row corresponding to a record of information; a plurality of columns intersecting said plurality of rows to define a plurality of cells, a cell being the basic unit of storage, each said column including an OID to identify each said column, at least one cell in a particular row including an **annotation** such that said **annotation** cell is fully integrated into said logical table; and performing an operation on said cell including said **annotation**.

18. The method of claim 16 wherein said **annotation** cell includes hypertext.

21. A method for storing and retrieving data in a computer system including a memory, a central processing unit and a display, said method including the steps of: configuring said memory according to a logical table, said logical table including; a plurality of rows, each said row including an object identification number (OID) to identify each said row, each said row corresponding to a record of information; a plurality of columns intersecting said plurality of rows to define a plurality of cells, each said column including an OID to identify each said column, at least one cell in a particular row including an **annotation** such that said **annotation** cell is fully integrated into said logical table; and performing an operation on said cell including said **annotation** wherein said **annotation** cell includes hypertext.

24. A method for storing and retrieving data in a computer system memory comprising the steps of: configuring said memory according to an extensible logical table having a plurality of intersecting rows and columns defining a plurality of cells, at least one of said columns including information to indicate an indexing method corresponding to

/

707/1+, 100+, 200+
707/500+

```
L5   ANSWER 1 OF 2  USPATFULL
PI     US 5644714  19970701
DETD   At step 36 the new clipping is stored locally (FIG. 2, memory 12). In
       the local storage, certain annotation, such as
       abstract and preview information, is stored to be accessible separately
       from the clipping itself (Step 38). Also, a copy of the new clipping,
       together with the annotation, is burst at step 40 to other file servers
       on the global network, to be stored and made available to subscribers
       connected to the other file servers on the global network, which also
       store the clipping and make it available to locally-connected
       subscribers. Finally, at step 42, local subscribers who have indicated
       an interest in the subject matter to which the new clipping pertains are
       notified by a standardized protocol. This notification is preferably in
       the form of E-mail which the subscribers may access
       at their leisure or pleasure.
DETD   If the request from a local subscriber is for one or more clippings, the
       requested clippings are retrieved at step 58 and sent to the subscriber
       at step 60. At step 58, assuming a request for downloading, another
       control path is initiated resulting in activation of an accounting
       procedure at step 62. At step 64 the request is logged and billing
       information is updated. Subscribers are periodically billed at step 70,
       which may be done by an E-mail service, just as in
       sending preview and listing information to subscribers.
DETD   Each sector transmitted through the maze of the massively parallel
       network is coded (sector #, destination, etc.), and as each sector
       arrives at client LAN station 225, it is recorded in memory according to
       prearranged addresses. After at least one sector is available, the
    playback may begin, by converting the available data to video
       signals and transmitting the signals to the video display. In most cases
       this is a CRT video tube, but that is not a requirement. As other types
       of displays (LCD, ELD, etc.) become more common for TV and high
       definition TV, the equipment at the client station can be updated to
       operate with the later apparatus.
CLM    What is claimed is:
       13. The video jukebox system of claim 1 wherein the file servers notify
       clients of available clippings by e-mail, and
       wherein the client instructs the system in downloading through e
       -mail.

       19. The method of claim 14 wherein, in the notifying clients step (d)
       the clients are notified by e-mail.


=> d pn, hit 2


L5   ANSWER 2 OF 2  USPATFULL
PI     US 5557515  19960917
DETD   A provision also exists within the Mailbox function to send intraoffice
    electronic mail (primarily administrative memos and
       the like). This function is preferably accessed through the "Wang.RTM.
       Office" automation program which is available when Wang.RTM. brand
       computers and peripherals are used throughout a claims office. This
       function is not, however, limited to Wang.RTM. brand equipment. One of
       skill in the art would be able to provide such a feature using any
       comparable hardware.
DETD   Referring to FIG. 26, when an outside call comes in for a staff member,
```

it passes through a Voice Bridge (e.g. Model ATT 7405 SET) where Voice Bridge Software which is integrated into the System, identifies the extension number dialed. The extension number is then passed to the Main CPU, where Wang.RTM. Office software (integrated into the System) identifies the staff member being called and finds a voice prompt associated with that staff member. Simultaneously, the call is routed to the Voice Front End Processor. The voice prompt is passed to the VFEP and the voice prompt message is played to the caller. When the voice prompt message is finished, the Main CPU records the caller verbal message in storage for subsequent **playback**. The Voice Bridge software then sends a message via Wang.RTM. Office **E-Mail** to the person being called that a message has been received in the "Voice Message Center." The user can then access the Voice Message Center from any touch tone phone and retrieve the message.

CLM What is claimed is:

10. The method accordingly to claim 2 comprising the additional steps of: annotating a selected scanned electronic image via an input to said processing instrumentality; saving said **annotations** on said **storage** means; merging said annotation and said selected image; and displaying said merged annotated image at one of said intelligent terminals.

said column, at least one cell includes an **annotation** such that said **annotation** cell is fully integrated into said extensible logical table; and performing an operation on said cell including said **annotation** wherein said **annotation** cell includes hypertext.

29. The method of claim 25 wherein said at least one row comprises a template for a structured **e-mail** message.

40. In a method for storing and retrieving information in a computer memory by configuring said memory according to a logical table having a plurality of rows, each said row corresponding to a record of information, and a plurality of columns intersecting said plurality of rows to define a plurality of cells, the improvement comprising: including an **annotation** in at least one cell in a first row, said **annotation** cell being fully integrated into said logical table; and performing an operation on said at least one cell including said **annotation**.

42. The method of claim 40 wherein said **annotation** cell includes hypertext.

45. A device for storing and retrieving data in a computer system memory comprising: an extensible logical table having a plurality of intersecting rows and columns defining a plurality of cells; at least one row includes a fields cell having references to a plurality of labeled columns at least one cell includes an **annotation** such that said **annotation** cell is fully integrated into said extensible logical table; and performing an operation on said cell including said **annotation** wherein said **annotation** cell includes hypertext.

50. The method of claim 46 wherein said at least one row comprises a template for a structured **e-mail** message.

61. A method for storing and retrieving data in a computer system memory comprising the steps of: configuring an extensible logical table in a computer system memory to include: a plurality of rows, each row corresponding to a record of information; a plurality of columns intersecting said plurality of rows to define a plurality of cells; and at least one cell in a first row including an **annotation** such that said **annotation** cell is fully integrated into said extensible logical table; and performing an operation on said cell including said **annotation**.

63. The method of claim 61 wherein said **annotation** cell includes hypertext.

68. The system of claim 64 wherein said at least one record comprises a template for a structured **e-mail** message.

79. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising: means for configuring said memory according to an extensible logical table, said extensible logical table including: a plurality of cells, each said cell having a first address segment and a second address segment; a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets, at least one cell in a first record including an **annotation** such that said **annotation** cell is fully integrated into said extensible logical table; and means for performing an operation on said cell including said **annotation**.

81. The system of claim 79 wherein said **annotation** cell includes hypertext.